## REMARKS

Claims 1, 4-9 and 11-19 remain pending in the application. Favorable reconsideration of the application is respectfully requested.

### *I. OBJECTION TO SPECIFICATION/REJECTION OF CLAIMS 1-19 UNDER 35 USC §112, 1ST ¶*

The Examiner initially objects to the specification as not including support for the various claim recitations identified on page 2 of the Office Action. Relatedly, the Examiner rejects claims 1-19 under 35 USC §112, first paragraph, as not being supported by the specification. Applicants respectfully request withdrawal of the objection/rejection for at least the following reasons.

The Examiner indicates that the applicants have not pointed out where the new (or amended) claims are supported. Applicants intended to address such matter in their response of February 16, 2007 in Section I(i) Claim Amendments. Nevertheless, applicants will elaborate further so as to make clear that the specification does in fact support such claim amendments.

#### *i. a CPU configured to judge whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code, independent of where the intermediate code is stored in the RAM;*

*FIG. 6*

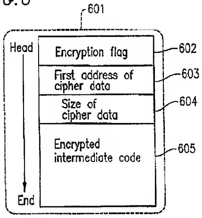


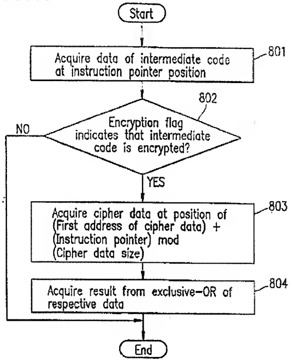
Fig. 6 of the present application exemplifies how intermediate code 601 stored in memory is identified as intermediate code or encrypted intermediate code based on the presence of an encryption flag 602 within the header. As is described in the present specification, for example:

From Specification:

*If the intermediate code is encrypted, data 0x01 is recorded as the encryption flag 602, and if the intermediate code is not encrypted, data 0x00 is recorded as the encryption flag 602.* (Spec., p. 19, lns. 8-18).

*At step 802, it is determined whether or not the intermediate code has been encrypted by referring to the value of the encryption flag (e.g., the encryption flag 602 as shown in Figure 6).* (Spec., p. 20, lns. 4-10).

FIG. 8


Start
Acquire data of intermediate code at instruction pointer position
801
802
Encryption flag indicates that intermediate code is encrypted?
NO
YES
Acquire cipher data at position of (First address of cipher data) + (Instruction pointer) mod (Cipher data size)
803
Acquire result from exclusive-OR of respective data
804
End

Thus, Fig. 8 of the present application exemplifies how the intermediate code stored in memory is obtained from memory, for example at an instruction pointer position within the memory. In step 802, it is determined whether or not the intermediate code has been encrypted by referring to the value of the encryption flag 602. If the intermediate code is judged to be the encrypted intermediate code based on the flag 602, the process proceeds to steps 803, 804 in order to decrypt the intermediate code. If the intermediate code is judged not to be encrypted in step 802 based on the flag 602, the decryption steps 803, 804 are skipped.

Thus, according to the present invention as disclosed in the present application as originally filed, <u>whether the intermediate code is judged as intermediate code or encrypted intermediate code is independent of where the intermediate code is stored in memory</u>, as recited in claims 1, 5 and 17. While the original specification does not expressly recite the exact language *"judg[ing] whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code, <u>independent of</u>*

*where the intermediate code is stored in the RAM"*, such language is clearly supported. The specification describes judging whether the intermediate code is encrypted or not based on the value of the encryption flag, not based on the location of the intermediate code as stored in the RAM. Consequently, the specification supports claim language wherein whether the intermediate code is encrypted or not is judged independent of where the intermediate code is stored in the RAM.

***ii. CPU ... configured to execute the interpreter execution program for interpreting the intermediate code; and configured to execute the interpreter execution program for decrypting and interpreting the encrypted intermediate code***

Again, applicants respectfully submit that the application includes support for such features:

From Specification:

*The execution means 120 [referring to Figure 1] includes a RAM 103 for storing the intermediate code 108, a ROM for storing the interpreter execution program 106 which is capable of interpreting the intermediate code 108, and a CPU 102 for controlling execution of the interpreter execution program 106. (Spec., p. 10, lns. 25-30).*

*[T]he CPU 102 in the LSI 101 executes the interpreter execution program 106 stored in the ROM 104. Then, the CPU 102 interprets and executes the intermediate code 108 stored in the RAM 103 by using the cipher data 107 stored in the ROM 104.* (Spec., p. 12, lns. 15-20).

Applicants respectfully submit that support for the subject claim language is self-explanatory based on the above passages taken directly from the original specification. Should the Examiner still feel that the claim language is unsupported, applicants respectfully request that the Examiner elaborate so that Applicants may better understand the Examiner's position.

***iii. wherein the CPU judges whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code based on header information included in the intermediate code...***

***wherein the header information is a flag.***

Applicants respectfully submit that such claim language also is supported by the original specification. As is discussed above in subsection (i), whether the intermediate code is encrypted or not is determined based on the value of the encryption flag 602 as shown in Figure 6). (Spec., p. 20, lns. 4-10). Such encryption flag 602 is included in header information as discussed in the specification:

> From Specification:
>
> *At step 307, at the head of the encrypted intermediate code string, 1 byte of data, 0x01, which functions as an encryption flag indicating that the intermediate code has been encrypted, the first address of the cipher data 107 on the ROM, and the size of the cipher data 107 are attached.* (Spec., p. 14, lns. 15-20) (Emphasis Added).

Again, therefore, applicants respectfully submit that the subject claim language is supported by the specification as originally filed. Applicants respectfully request withdrawal of the objection/rejection.

## *II. REJECTION OF CLAIMS 1, 12 AND 15-19 UNDER 35 USC §103(a)*

Claims 1, 12 and 15-19 stand rejected under 35 USC §103(a) based on *Westheimer et al.* in view of *Buerkle et al.* Applicants respectfully request withdrawal of the rejection for at least the following reasons.

*i. Judgment Independent of Where Code Stored*

Applicants previously pointed out how claim 1 recites, *inter alia*, that the CPU is "configured to *judge whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code*, independent *of where the intermediate code is stored in the RAM*. Amended claim 5 and new claim 17 recite similar features. Neither *Westheimer et al.* nor *Buerkle et al.*, whether taken alone or in combination, teach or suggest such features.

The Examiner responds to applicants' argument at page 11 of the Office Action, noting that:

> *In response, the examiner respectfully notes that, regardless of where the code is stored, the combination discloses that judgment can be made (Westheimer, 2:54-57). Thus, "independently" as claimed.* (O.A., p. 11).

The text cited by the Examiner reads as follows:

> *The transformation enable circuit is initially activated by a memory boundary operation code which will normally be one of the first instructions in an encoded program.* (Westheimer, Col. 2, lns. 54-57).

Applicants note, however, in the very same paragraph to which the Examiner refers to in Westheimer, it is pointed out that:

> *Once activated, the transformation enable circuit monitors the flow of data and addresses between the central processing unit of the computer and the computer main memory. Data and addresses which fall within a segment of memory defined by the memory boundary operation code undergo transformation, while data and addresses which fall outside said segment remain unaffected.* (Col. 2, lns. 58-65).

Applicants respectfully submit that it is clear from the above-cited text that Westheimer is *not* teaching "a CPU configured to judge whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code, independent of wherein the intermediate code is stored in the RAM" as recited in the claims. Westheimer expressly teaches that whether an instruction is judged encoded or not *is based on where the code is stored in the memory* (i.e., whether the code is stored within a segment of memory defined by the memory boundary operation code, or outside of said segment). Judgment based on where the code is stored in the RAM, as taught in Westheimer, can hardly be viewed as judgment independent of where the code is stored in RAM, as claimed.

Consequently, applicants respectfully submit that one having ordinary skill in the art would not construe Westheimer to teach or suggest "a CPU configured to judge whether intermediate code obtained from the RAM is the intermediate code or the encrypted intermediate code, independent of where the intermediate code is stored in the RAM" as recited in claims 1, 5 and 17.[1]

Applicants therefore respectfully submit that the Examiner's continued reliance on Westheimer is misplaced and the rejection should be withdrawn.

*ii. Westheimer et al. and Buerkle et al. Do Not Teach/Suggest Intermediate Code*

Applicants also previously argued that neither Westheimer et al. nor Buerkle et al. teach or suggest storing both intermediate code and encrypted intermediate code in RAM. Applicants pointed out that Westheimer relates to storing *operation* code (i.e., machine code instructions) and encrypted *operation* code in RAM, not *intermediate* code as claimed.

The Examiner responds by stating:

---

[1]Applicants incorporate herein by reference the detailed description of Westheimer as set forth in their previous response filed on February 16, 2007, at pages 9-11. Such description provides a detailed explanation.

> *In response to applicants arguments against the references individually, one cannot show nonobviousness by attaching references individually where the rejections are based on combinations of references (citations omitted). Regarding the applicant's above argument, the examiner respectfully points out that claims 1 and 12 stand rejected in view of Westheimer and Buerkle.*
>
> *In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., Basic, Java, PASCAL, etc.) are not recited in the rejected claims.* (O.A., p. 10).

Regarding Westheimer, applicants referred to Basic, Java, PASCAL, etc., merely as examples of intermediate code, and to exemplify the distinction between intermediate code and machine code as in Westheimer. Westheimer does not teach or suggest the storage of intermediate code and encrypted intermediate code as recited in the claims.

Regarding Buerkle, this reference also fails to teach or suggest the storage of intermediate code and encrypted intermediate code in memory as recited in claims 1, 5 and 17. Thus, even if the references are viewed in combination as suggested by the Examiner, the references still do not teach or suggest the claimed invention. Neither reference teaches storing intermediate code and encrypted intermediate code in memory, and thus nor does the combination. The fact that the claims do not recite Basic, Java, PASCAL, etc. is not material to applicants' argument.

In case the Examiner may find it helpful, attached are three different articles, "A RISC Interpreter for the C Programming Language" by Davidson et al., "Techniques for Obtaining High Performance in Java Programs" by Kazi et al., and "Comparing Java Implementations for Linux" by Hirsch. These articles explain in more detail the differences between intermediate code and operation code. Namely, intermediate code is not dependent upon the CPU specification, whereas operation or machine code is dependent upon the CPU specification.

*iii. No Interpretation Execution Program Stored in ROM*

The Examiner admits that Westheimer does not disclose that the CPU operates using an "interpreter execution program" stored in ROM. However, the Examiner submits that Buerkle teaches such feature and thus makes up for the deficiencies of Westheimer. Applicants respectfully submit that Buerkle does not teach an "interpreter execution program" stored in ROM, and rather teaches away from such feature.

The Examiner asserts that Buerkle teaches that LSI processors utilize an "interpreter execution program" [microprogram] to allow a CPU to process intermediate code to allow a CPU to process intermediate code [macroinstructions] according to the opcodes of the instructions. The Examiner indicates that Buerkle teaches that prior art discloses LSIs as storing the "interpreter execution program" in a ROM (Buerkle, "Description of the Related Art"). (See, O.A., ¶ bridging p. 4-5).

Specifically, the Examiner indicates that it would have been obvious to one of ordinary skill in the art to recognize the need for an "interpreter execution program" stored in a ROM as recited in claims 1, 6 and 17 to allow a CPU to control the execution of an "intermediate code", and thus follow the LSI processor design teachings of Buerkle within the LSI processor system of Westheimer. The Examiner suggests this would have been obvious because one of ordinary skill in the art would have been motivated to practically implement the features known in the prior art to be included within LSI processor systems. (See, O.A., p. 5, lns. 4-10).

Applicants respectfully disagree with such conclusion. In fact, comparing the disclosure in the section "Description of the Related Art" in Buerkle with the disclosure in the "Description of the Related Art" in the present application, it is clear that the respective disclosures refer to the same problem with the conventional art.

Particularly, in the present application is taught that:

> *[i]n a conventional LSI, software used for <u>performing its operation is previously stored in a memory such as a ROM</u> or the like. When the LSI performs its operation, a <u>CPU reads software from the ROM</u>. In the above conventional LSI, <u>only software previously stored in the ROM can be executed, and it is not permitted to freely make a modification to software stored in the ROM</u>. Thus, <u>such an LSI cannot be used with various disc apparatuses available from a plurality of manufacturers</u>,... . Furthermore, modifying specifications or adding functions, a need for which may arise in the process of development of a disc apparatus, cannot be readily satisfied..." (Emphasis added; See, Spec., ¶ bridging p. 1-2).*

In the section "Description of the Related Art" in Buerkle, it is disclosed that:

> *Generally, a processor performs a series of operations upon digital information in an execution unit according to a stored program of instructions. These instructions are often termed "macroinstructions" in order to avoid confusion with the "microinstructions" contained in the control store of the processor. <u>The microinstructions are often grouped into microinstruction routines in order to perform various macroinstructions</u>. The grouped microinstruction routines constitute the microprogram of the processor. (Emphasis added; Col. 1, lns. 30-39).*
>
> *Prior art processors use microprogrammed instructions to control internal circuitry in response to macro-instructions. These processors use table look-up approaches to map a macroinstruction into the appropriate microinstruction routines/microprogram. ... Each micro-branch is accessed by an address formed directly from the operation-code (op-code) of the macroinstruction. <u>In addition, the micro-branch entries are physically remote from the processor in firmware, a ROM</u>, to permit flexibility during the manufacturing process. <u>One disadvantage of this approach is that entries in the remote table cannot be directly manipulated by the processor. This eliminates flexibility to make changes caused by development and later engineering changes</u>. A second disadvantage results from the micro-branching process which requires extra operating cycles when the branching operations are executing. <u>These extra cycles result in considerable delay between the time a processor work sequence is initiated by a macroinstruction and the decode of the first useful microinstruction routine</u>. (Emphasis added; Col. 1, lns. 40-65).*

In other words, Buerkle teaches that "microinstructions" are stored in ROM, and that "macroinstructions" are effective groupings of microinstruction routines that constitute the microprogram of the processor. Buerkle does not appear to teach that an interpreter execution program is stored in ROM to interpret intermediate code, and to decrypt and interpret encrypted intermediate code. Rather, macroinstructions are performed by mapping (using table look-up approaches) such macroinstructions into appropriate microinstruction routines/microprogram.

As mentioned above, this is similar to the "Related Art" as described in the present application. In both the present application and Buerkle, this storing of microinstructions in ROM is described as being disadvantageous.

Therefore, even if the microprogram of Buerkle were to be interpreted as "an interpreter execution program", Buerkle seems to actually teach away from such microprogram being stored in ROM (see, particularly, Col. 1, lns. 51-65).

Furthermore, applicants respectfully submit that the microprogram is not the "interpreter execution program" of the present claimed invention, contrary to the Examiner's assertions. Rather, the microprogram represents the command control string to be executed by the control section, so that the microprogram more clearly resembles the intermediate code of the present claims.

The "program that interprets" the microprogram includes the table look-up approaches that map the macroinstruction into the appropriate microinstruction routines/microprogram.

In summary, a person of ordinary skill in the art studying Buerkle would not be motivated to store an interpreter executing program in ROM, because (1), even if the "microprogram" of Buerkle were to be interpreted as an "interpreter executing program", Buerkle teaches away from storing the microprogram in ROM due to the disadvantages as highlighted above, (2) the "microprogram" of Buerkle is *not* an "interpreter executing program" (since it does not interpret the intermediate code; but is actually more representative of the intermediate code itself), and (3) Buerkle does not actually teach or suggest a program used for interpreting intermediate code, wherein the interpreter execution program is stored in ROM.

Therefore, Buerkle does not teach the feature "a ROM for storing an *interpreter execution program* that is configured to interpret the intermediate code, and to decrypt and interpret the encrypted intermediate code, as recited in claims 1, 6 and 17.

For at least reasons (i) thru (iii) discussed above, applicants respectfully submit that Westheimer and Buerkle, taken alone or in combination, do not teach or suggest the present invention as claimed. Applicants respectfully request that the rejection be withdrawn.

## ***III. REJECTION OF CLAIMS 4-14 UNDER 35 USC §103(a)***

Claims 4-14 stand rejected under 35 USC §103(a) based on *Westheimer et al.* in view of *Buerkle et al.*, and further in view of *Hagiwara et al.* Applicants respectfully request withdrawal of this rejection for at least the following reasons.

Regarding independent claim 5, this claim recites features similar to claim 1 as discussed above. Accordingly, claim 5 may be distinguished over the teachings of *Westheimer et al.* and *Buerkle et al.* for at least the same reasons expressed above. Moreover, *Hagiwara et al.* does not make up for the above-discussed deficiencies in *Westheimer et al.* and *Buerkle et al.*

The remaining claims represent dependent claims that depend from either claim 1 or claim 5 discussed above, and may be distinguished for at least the same reasons.

## *IV. REJECTION OF CLAIMS 1-19 UNDER 35 USC §103(a)*

Claims 1-19 now stand rejected under 35 USC §103(a) based on *CN2045628U* in view of *CN1245926A*. Applicants respectfully request withdrawal of this rejection for at least the following reasons.

In rejecting the claims, the Examiner points to the rationale expressed in the Office Action dated July 9, 2004 in relation to the corresponding Chinese application.

The Chinese Examiner asserts that the difference between claim 1 and *CN2045628U* is the content stored in the RAM and ROM. The Chinese Examiner asserts that the content stored in storage can be voluntarily appointed based on particular application, which does not need creative effort for the person skilled in the art. The Chinese Examiner further asserts that the technological problem to be solved by claim 1 and *CN2045628U* is to integrate RAM, ROM and CPU into one chip.

Applicants note, on the other hand, even though *CN2045628U* does disclose that RAM, ROM and CPU are integrated into one chip, the RAM does not store an intermediate code and the ROM does not store an interpreter execution program which is capable of interpreting the intermediate code as claimed. *CN1245926A* does not make up for such deficiencies.

Accordingly, applicants respectfully request withdrawal of the rejection.

## *V. CONCLUSION*

Accordingly, all claims 1-19 are believed to be allowable and the application is believed to be in condition for allowance. A prompt action to such end is earnestly solicited.

Should the Examiner feel that a telephone interview would be helpful to facilitate favorable prosecution of the above-identified application, the Examiner is invited to contact the undersigned at the telephone number provided below.

Should a petition for an extension of time be necessary for the timely reply to the outstanding Office Action (or if such a petition has been made and an additional extension is necessary), petition is hereby made and the Commissioner is authorized to charge any fees (including additional claim fees) to Deposit Account No. 18-0988.

Respectfully submitted,

RENNER, OTTO, BOISSELLE & SKLAR, LLP

/Mark D. Saralino/
Mark D. Saralino
Registration No. 34,243

DATE: August 7, 2007

The Keith Building
1621 Euclid Avenue
Nineteenth Floor
Cleveland, Ohio 44115
(216) 621-1113

B:\GEN\YAMA\Yamap797\yamap797amendmenafterfinal5.wpd